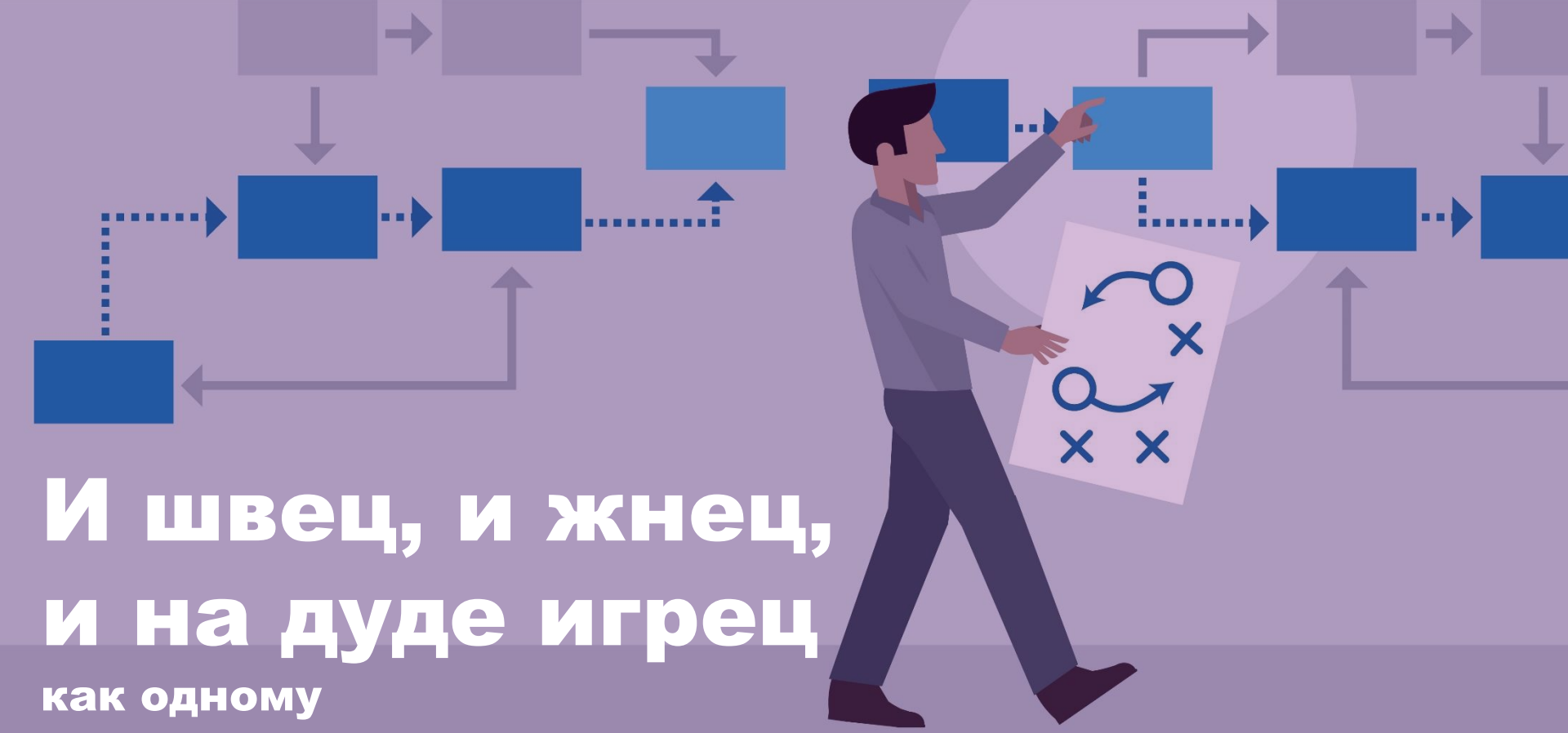


И швец, и жнец, и на дуде игрец

как одному докатить модель в прод

Селезнев Артем, 24 июня, 2019

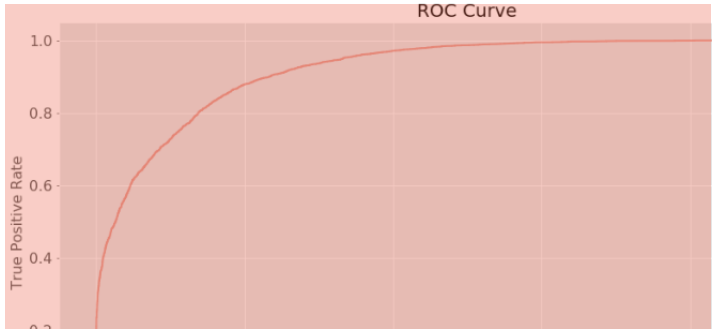
1 Jupyter notebook и Git

Jupyter notebook и Git. В чем состоит проблема для DS?

- Jupyter notebook – это JSON файл для GIT

```
    auc = sklearn.metrics.auc(fpr, tpr)
    return auc
```

```
In [19]: calculate_score_on_dataset_and_show_graph(X_test, y_test, model)
```




```
{
 "data": {
  "image/png": "iVBORw0KGgoAAAANSUhEUgAAAfgAAAH0CAYAAADR6j8eAAAABHNCSVQICAgIfAhkiAAAAAl
  "image/png": "iVBORw0KGgoAAAANSUhEUgAAA8IAAAHCAYAAADR6GT3AAAABHNCSVQICAgIfAhkiAAAAAl
  "text/plain": [
   "<Figure size 576x576 with 1 Axes>"
   "<Figure size 1152x576 with 1 Axes>"
  ]
 },
 "metadata": {},
```

Jupyter notebook и Git. Возможные решения

1. Cell – All output – Clear

Jupyter notebook и Git. Возможные решения

1. Cell – All output – Clear

2. Extention Tools – File save hooks

   ContentsManager.pre_save_hook / post_save_hook (contents_manager = cn)

Jupyter notebook и Git. Возможные решения

1. Cell – All output – Clear

2. Extention Tools – File save hooks

   ContentsManager.pre_save_hook / post_save_hook (contents_manager = cn)

3. Заменить pre_save_hook

```
def remover(model, **kwargs):
    if model['type'] != 'notebook':
        return

    for cell in model['content']['cell']:
        if cell['cell_type'] == 'code':
            cell['execution_count'] = None
            cell['outputs'] = []
```

Jupyter notebook и Git. Возможные решения

4. NBFormat lib

```
import sys
import nbformat

notebook = nbformat.read(sys.stdin)

for cell in notebook.cells:
    #the same code …

nbformat.write(notebook, sys.stdout)
```

Jupyter notebook и Git. Возможные решения

4. NBFormat lib

```
import sys
import nbformat

notebook = nbformat.read(sys.stdin)

for cell in notebook.cells:
    #the same code …

nbformat.write(notebook, sys.stdout)
```

5. Nbstripout / NBDime

Jupyter notebook и Git. Результат

```
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'r', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts, 
36 ax.add_patch(poly)
37
   (...)
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
```

```
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'r', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts, 
36 ax.add_patch(poly)
37
   (...)
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
```

```
27 fig, ax = plt.subplots
28 plt.plot(x, y, 'g', li
29 plt.ylim(ymin=0)
30
31 # Make the shaded regi
32 ix = np.linspace(a, b)
33 iy = func(ix)
34 verts = [(a, 0)] + lis
35 poly = Polygon(verts, 
36 ax.add_patch(poly)
37
   (...)
41 plt.figtext(0.9, 0.05,
42 plt.figtext(0.1, 0.9,
```

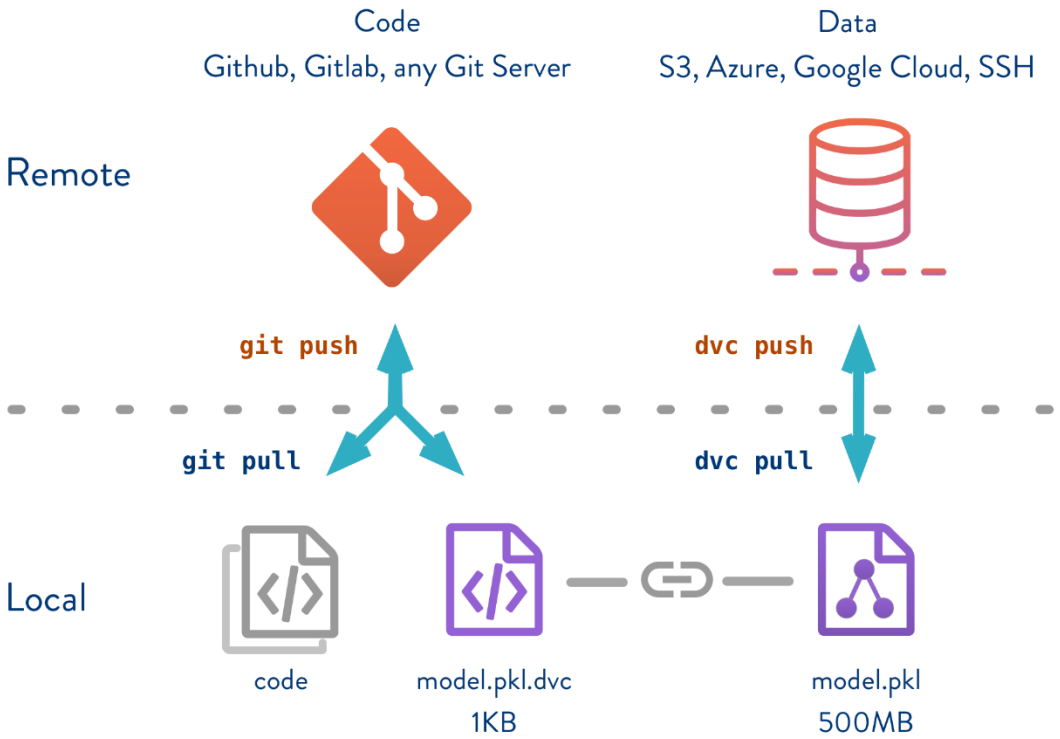
2 ML проект с данными Data Version Control

DVC. Хранение данных – проблема?

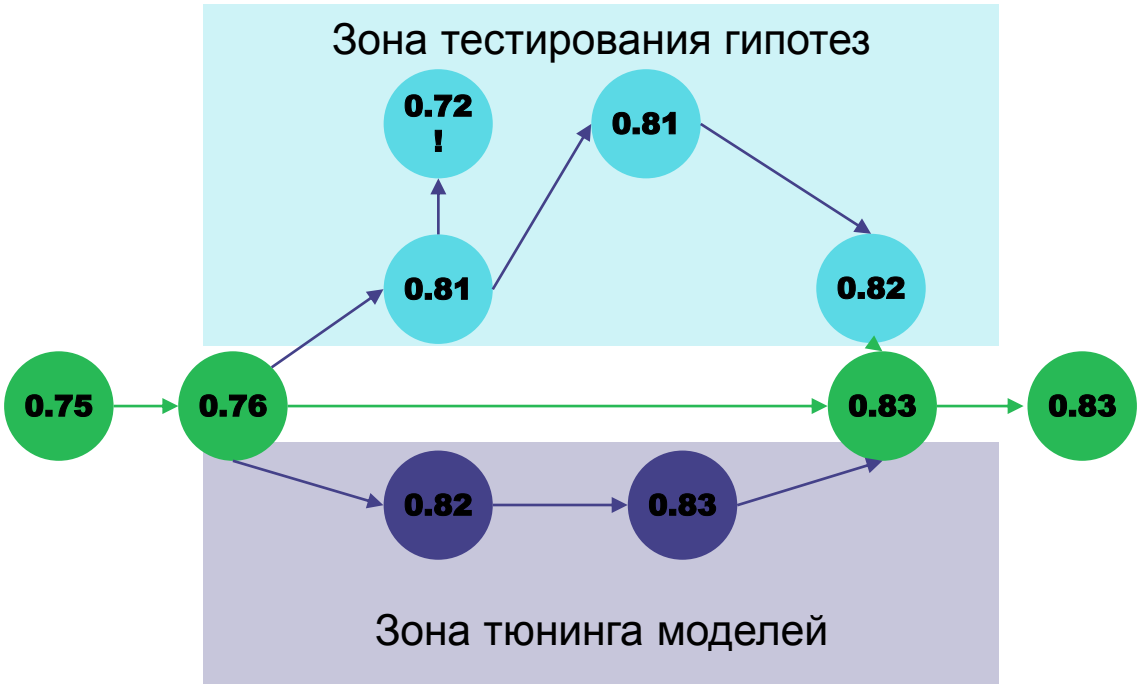
```
/usr/lib64/python3.4/site-packages/pandas/core/frame.py in extract_index(data)
   5542                 lengths = list(set(raw_lengths))
   5543                 if len(lengths) > 1:
-> 5544                     raise ValueError('arrays must all be same length')
   5545
   5546                 if have_dicts:

ValueError: arrays must all be same length
```

```
ValueError: '30_lift_set_1.csv' is not in list
```

DVC. Хранение данных – это решение!



DVC. Хранение данных – проблема. Почему?



1. Сохранять метрики
2. Версии моделей
3. Версии данных
4. Версии `pipelines`

DVC в Python оболочке

1. Python magic:
   __enter__
   __exit__

```python
from dvc_controller import Controller

with Controller('path', 30, 'methods') as cntr:
    ...
```

```
Dvcfile                [----] 13 L:[  1+ 1   2
md5: 82a403c988fa9f6795123cc44857ec1b
cmd: python3 wine/code.py
wdir: .
deps:
- md5: 7e973f172ecfdde98cc74c495b73d63b
  path: wine/code.py
outs:
- md5: c838251ba1a08f786e33d4e90afb7286
  path: wine/eval.txt
  cache: true
  metric: true
  persist: false
```

DVC в Python оболочке

1. Python magic:
   __enter__
   __exit__

2. Python + GraphJS

```
from dvc_controller import Controller

with Controller('path', 30, 'methods') as cntr:
    cntr.show()
```

DVC в Python оболочке

1. Python magic:
   __enter__
   __exit__

2. Python + GraphJS

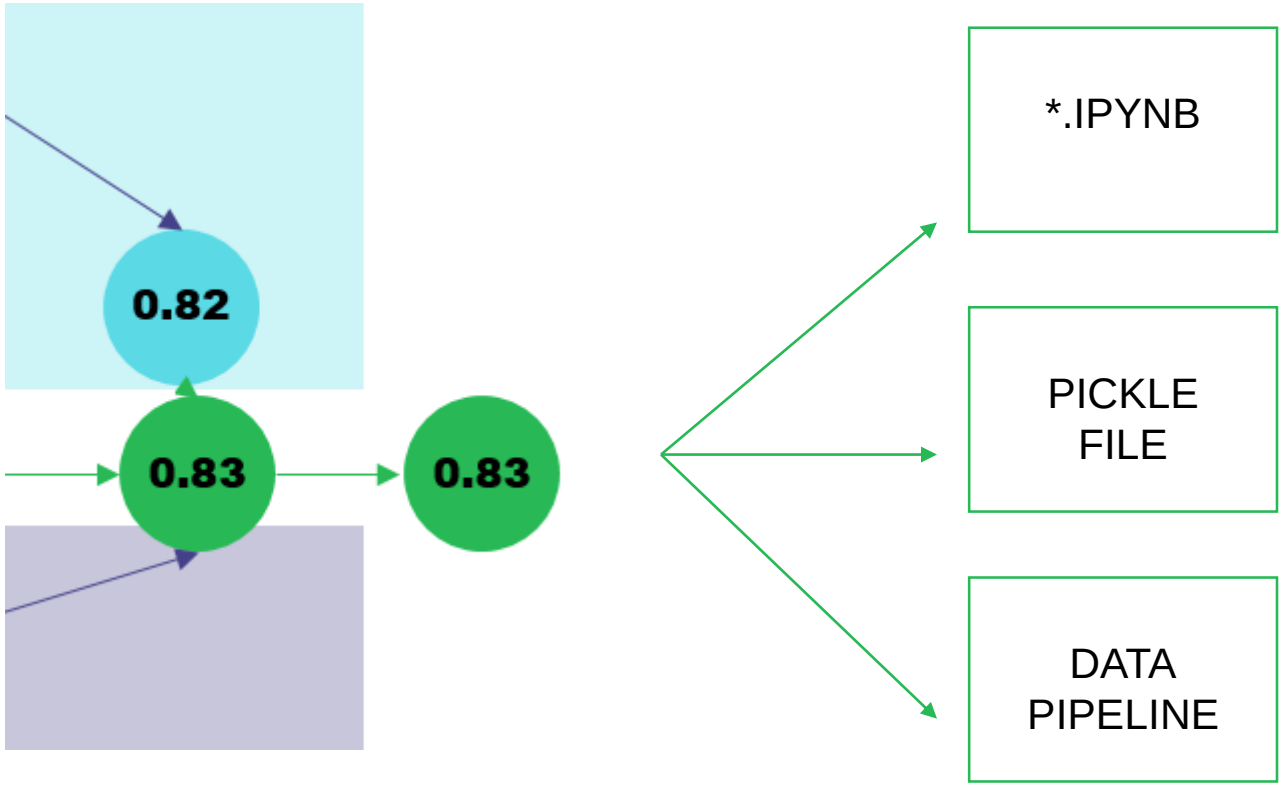
3. Other methods

```
from dvc_controller import Controller

with Controller('path', 30, 'methods') as cntr:
#add and commit to DVC
cntr.commit('message', 'tags')
#reproduce
cntr.repro('branch', 'date')
#change branch
cntr.branc('branch name')
#entire DB if you don't use local files
cntr.entire_db('link')
```

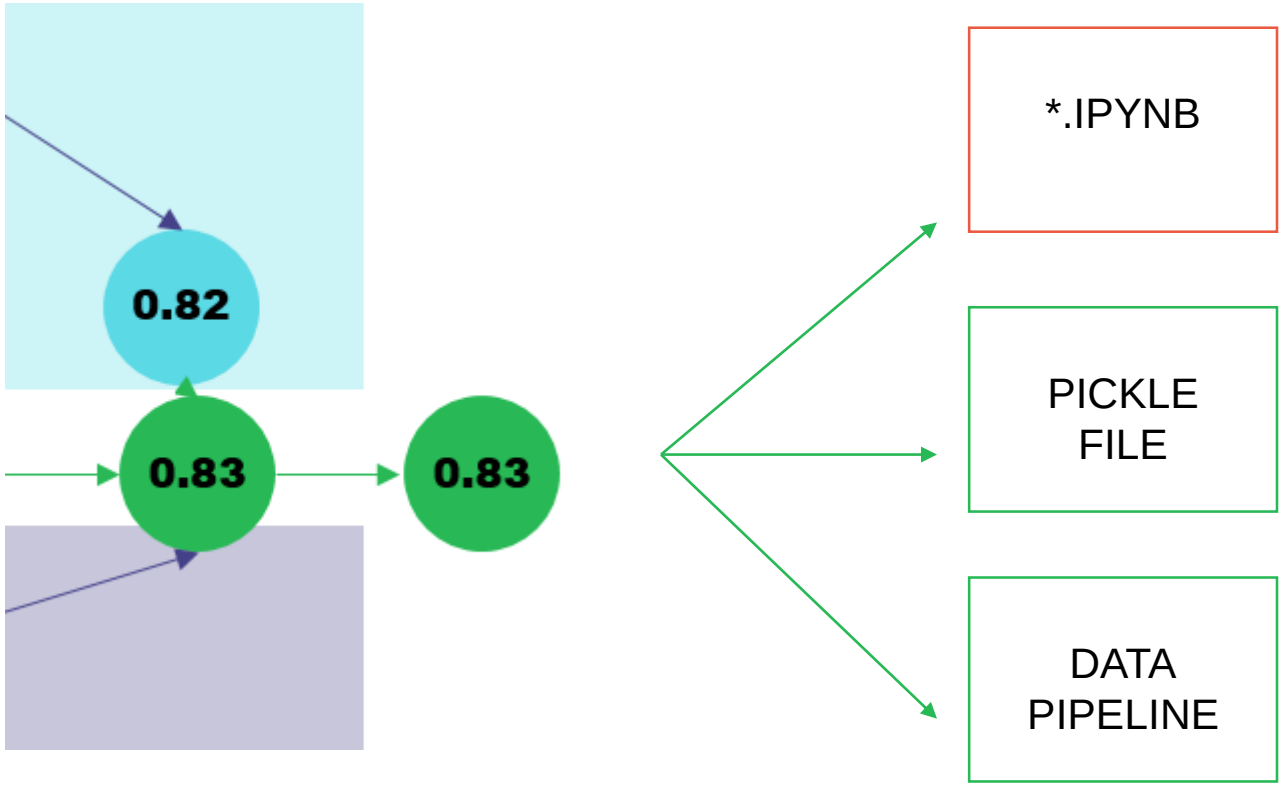
```
➢ dvc run -d code/note.ipynb -code/tune.ipynb \
          -o data/train.csv -m code/eval.txt \
          python3 code/note.ipynb \
                  code/tune.ipynb
```

3 Строим платформу

От эксперимента не осталось ничего



От эксперимента не осталось ничего



Data pipeline

Luigi

```
import luigi
from luigi import Task
from luigi.contrib.sqla import CopyToTable

class ETL(CopyToTable, Task):
    …
    columns = [
        (["id", Integer], {"primary_key": True}),
        (["share", String], {})
    ]
    #define a table
    def process(self):
        SQL = "QUERY"
        def run(self):
            with psycopg2.connect(connect_str) as c:
                engine.execute(sql)
                 …
        def output(self):
            with psycopg2.connect(connect_str) as c:
                engine.execute(new_sql)
                 …
```

Bonobo

```
import bonobo
import bonobo_sqlalchemy

class ETL:
      …
      def get_etl(**options):
                graph = bonobo.Graph()
                graph.add_chain(bonobo_sqlalchemy.\
                     Select("QUERY"),
               … #your process
              bonobo_sqlalchemy.InsertOrUpdate('Out_Table')
              )

              return graph
```

Data pipeline

Luigi

```
import luigi
from luigi import Task
from luigi.contrib.sqla import CopyToTable

class ETL(CopyToTable, Task):
    …
    columns = [
        (["id", Integer], {"primary_key": True}),
        (["share", String], {})
    ]
    #define a table
    def process(self):
        SQL = "QUERY"
        def run(self):
            with psycopg2.connect(connect_str) as c:
                engine.execute(sql)
                 …
        def output(self):
            with psycopg2.connect(connect_str) as c:
                engine.execute(new_sql)
                 …
```

Bonobo

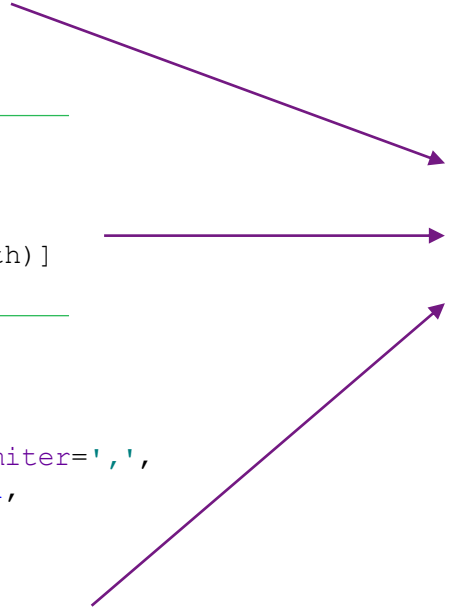
```
import bonobo
import bonobo_sqlalchemy

class ETL:
      …
      def get_etl(**options):
                graph = bonobo.Graph()
                graph.add_chain(bonobo_sqlalchemy.\
                     Select("QUERY"),
               … #your process
              bonobo_sqlalchemy.InsertOrUpdate('Out_Table')
              )

              return graph
```

Data pipeline

```
#SOME MARKER
def extract():
    ...
    result = engine.execute(sql)
    return result
```

```
#SOME MARKER
def transform(result_sql):
    path = 'Some path to a folder'
    files = [one for one in listdir(path)]
    ...
```

```
#SOME MARKER
def load(csv):
    ...
    data = data = genfromtxt(csv, delimiter=',',
                             skip_header=1,
converters={0: lambda s: str(s)})
    for i in data:
        record = Name(**{
            'col_name': i[n]
        })
        s.add(record)
    s.commit()
    ...
```



???

Data pipeline

```
#SOME MARKER
def extract():
    ...
    result = engine.execute(sql)
    return result
```

```
#SOME MARKER
def transform(result_sql):
    path = 'Some path to a folder'
    files = [one for one in listdir(path)]
    ...
```

```
#SOME MARKER
def load(csv):
    ...
    data = data = genfromtxt(csv, delimiter=',',
                             skip_header=1,
converters={0: lambda s: str(s)})
    for i in data:
        record = Name(**{
            'col_name': i[n]
        })
        s.add(record)
    s.commit()
    ...
```

```
import bonobo

graph = bonobo.Graph(
    result = engine.execute(sql),
    path = 'Some path to a folder'
    files = [one for one in listdir(path)]
    ...
    #execute files
    ...,
    data = data = genfromtxt(csv, delimiter=',',
                             skip_header=1,
                             converters={0:lambda s: str(s)})
    for i in data:
        record = Name(**{'col_name': i[n]})
        s.add(record)
    s.commit()


if __name__ == "__main__":
    bonobo.run(graph)
```

Создавай процессы
Начинающиеся с тебя

fb: /seleznev.artem.info
telegram: @SeleznevArtem